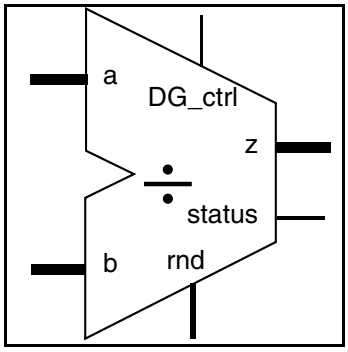
# DW_fp_div_DG

## Floating-Point Divider with Datapath Gating

Version, STAR, and myDesignWare Subscriptions: IP Directory

## Features and Benefits

Revision History




- Has the same functionality as DW_fp_div when the component is in normal operation
- Consumes less dynamic power than DW_fp_div when disabled (inputs are active, but the output is not being used)
- The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- Hardware for denormal numbers of IEEE 754 standard is provided.
- Configurable to be fully compliant with the IEEE Std 754-1985 standard
- Configurable for NaN representation compatible with the IEEE Std 754-2008 standard (controlled by the *ieee_compliance* parameter)
- Faithful rounding with 1 ulp error is supported by parameter
- DesignWare datapath generators are employed for better power and QoR

## Description

DW_fp_div_DG is a floating-point divider that divides two floating-point operands: `a` by `b` to produce a floating-point quotient, `z`. Also, a control input (`DG_ctrl`) can disable the component to reduce dynamic power consumption when the component is not in use and inputs are still active.

**Table 1-1 Pin Description**

| Pin Name | Width | Direction | Function |
|---|---|---|---|
| a | (*sig_width* + *exp_width* + 1) bits | Input | Dividend |
| b | (*sig_width* + *exp_width* + 1) bits | Input | Divisor |
| rnd | 3 bits | Input | Rounding mode; supports all rounding modes described in the *Datapath Floating-Point Overview*.<br>The `rnd` port takes effect only when *faithful_round* = 0. |
| z | (*sig_width* + *exp_width* + 1) bits | Output | Quotient of A/B |

**Table 1-1 Pin Description (Continued)**

| Pin Name | Width | Direction | Function |
|---|---|---|---|
| status | 8 bits | Output | ■ Status flags corresponding to `z`; for details, see STATUS Flags in the *Datapath Floating-Point Overview*<br>■ status[6]: Underflow before rounding (UBR) status flag<br>When enabled by the *en_ubr_flag* parameter, this flag indicates when the absolute value of a non-zero result, computed as though both the exponent range and the precision of the significand were unbounded, would lie strictly between ±MinNorm (minimum normalized value representable in the FP format defined by *sig_width* and *exp_width*).<br>■ status[7]: Indicates divide-by-zero operation |
| DG_ctrl | 1 bit | Input | Datapath gating control<br>■ 0: Component is disabled<br>■ 1: Normal component operation<br>See also "Datapath Gating Control with DG_ctrl" on page 4. |

**Table 1-2 Parameter Description**

| Parameter | Values | Description |
|---|---|---|
| sig_width | 2 to 253 bits | Word length of fraction field of floating-point numbers `a`, `b`, and `z` |
| exp_width | 3 to 31 bits | Word length of biased exponent of floating-point numbers `a`, `b`, and `z` |
| ieee_compliance | 0, 1, or 3<br>Default: 0 | Level of support for the IEEE Std 754 standards:<br>■ 0: No support for NaNs and denormals; NaNs are considered infinities and denormals are considered zeros<br>■ 1: Fully compliant with the IEEE Std 754-1985 standard, including support for NaNs and denormals<br>■ 2: Reserved<br>■ 3: Fully compliant with the IEEE Std 754-1985 standard plus NaN representation that matches the IEEE Std 754-2008 standard[a]<br>For details, see Compatibility with IEEE Std 754 Standards in the *Datapath Floating-Point Overview* |

**Table 1-2 Parameter Description**

| Parameter | Values | Description |
|---|---|---|
| faithful_round | 0 or 1<br>Default: 0 | Choose either a specific rounding mode (set by `rnd`) or a general rounding mode that allows maximum 1 ulp error<br>■ 0: Rounding mode is specific, as set by the `rnd` port; this choice increases the size of the resulting implementation.<br>■ 1: Rounding mode is general and, for *sig_width* ≤ 23, allows a maximum of 1 ulp error[b]; this choice decreases the size of the resulting implementation.<br>When *faithful_round* = 1, note the following:<br>- The inexact status flag in the output is not meaningful.<br>- The UBR status flag (`status`[6]) is always 0 (disabled).<br>- The other status flags will match one of the possible outputs for the calculation when *faithful_round* = 0. |
| en_ubr_flag | 0 or 1<br>Default: 0 | Controls when the UBR flag in `status`[6] is enabled<br>■ 0: The UBR status flag `status`[6] is disabled (the value of this flag is always 0).<br>■ 1: When this parameter is 1 and *faithful_round* = 0, `status`[6] carries the value of the UBR flag. If *faithful_round* = 1, even when *en_ubr_flag* = 1, the UBR flag in `status`[6] bit is always 0. |

a. Propagating payload information to the output during the NaN process, which is an optional feature specified in the IEEE Std 754-2008 standard, is not supported.
b. When *faithful_round* = 1 and *sig_width* > 23, the result can exceed 1 ulp for some corner cases. Configurations tested for this parameter range do not show errors larger than 2 ulps.

**Table 1-3 Synthesis Implementations**

| Implementation Name | Function | License Feature Required |
|---|---|---|
| rtl | Datapath gating close to the main inputs (Digit-recurrence Method) | ■ DesignWare (P-2019.03 and later)<br>■ DesignWare-LP (before P-2019.03) |
| str | Datapath gating close to the main inputs (Newton-Raphson Method) | ■ DesignWare (P-2019.03 and later)<br>■ DesignWare-LP (before P-2019.03) |
| rtl2[a] | Datapath gating to allow late arrival time of DG_ctrl signal | ■ DesignWare (P-2019.03 and later)<br>■ DesignWare-LP (before P-2019.03) |
| str2[a] | Datapath gating to allow late arrival time of DG_ctrl signal | ■ DesignWare (P-2019.03 and later)<br>■ DesignWare-LP (before P-2019.03) |

a. By default, the rtl2/strl2 implementation is used for synthesis (see "Datapath Gating Control with DG_ctrl" on page 4).

**Table 1-4 Simulation Models**

| Model | Function |
|---|---|
| DW02.DW_FP_DIV_DG_CFG_SIM | Design unit name for VHDL simulation |
| dw/dw02/src/DW_fp_div_DG_sim.vhd | VHDL simulation model source code |
| dw/sim_ver/DW_fp_div_DG.v | Verilog simulation model source code |

## Datapath Gating Control with DG_ctrl

For DW_fp_div_DG and other combinational components that have the datapath gating feature, the `DG_ctrl` port is provided to control datapath gating.

When `DG_ctrl` = 1, the component behaves as expected according to activity on the input ports.

When `DG_ctrl` = 0:

- The component is disabled and internal gates are totally or partially isolated to block propagation of switching activity inside the component. This makes the component less sensitive to switching activity on the main ports and reduces dynamic power consumption.
- Values at the output ports are not defined.
- Simulation models set 'X' values at the output ports.

The implementations for DW_fp_div_DG (see Table 1-3 on page 3) perform datapath gating differently:

- The rtl/str implementation places datapath gating as close as possible to the input ports to maximize dynamic power savings when `DG_ctrl` = 0. However, if the `DG_ctrl` signal arrives later than the data inputs, timing is degraded and area is increased to recover timing, which can increase power.
- The rtl2/str2 implementation places datapath gating near the middle of the component. This approach is less sensitive to the arrival time of `DG_ctlr` and has a better chance of meeting timing and still providing dynamic power savings.

By default, the synthesis tool uses the rtl2/str2 implementation, but you can override that. If timing constraints are loose or you know that the signal driving the `DG_ctrl` port arrives at the same time as other input signals, greater power savings can be attained by using the rtl/str implementation. You can make the override on a global level or on a case-by-case basis, as explained next.

To use the rtl/str implementation globally, you can disable the rtl2/str2 implementation as follows:

- Design Compiler (before version P-2019.03):

```
set_dont_use {dw_minpower.sldb/DW_fp_div_DG/rtl2
    dw_minpower.sldb/DW_fp_div_DG/str2}
```

- Design Compiler (P-2019.03 and later)

```
set_dont_use {dw_foundation.sldb/DW_fp_div_DG/rtl2
    dw_foundation.sldb/DW_fp_div_DG/str2}
```

- Fusion Compiler:

```
set_synlib_dont_use {dw_foundation/DW_fp_div_DG/rtl2
    dw_foundation/DW_fp_div_DG/str2}
```

To use the rtl or str implementation for specific instantiated components, use the set_implementation command. For example:

```
set_implementation U1 rtl
```

## Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If an invalid rounding mode has been detected on `rnd`, the following message is displayed:

  ```
  WARNING: <instance_path>:
      at time = <timestamp>: Illegal rounding mode.
  ```

  To suppress this message, use the DW_SUPPRESS_WARN macro explained earlier.

## Related Topics

- Datapath Floating-Point Overview
- DesignWare Building Blocks User Guide

## HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_Foundation_comp.all;

entity DW_fp_div_DG_inst is
      generic (
        inst_sig_width : POSITIVE := 23;
        inst_exp_width : POSITIVE := 8;
        inst_ieee_compliance : INTEGER := 0;
        inst_faithful_round : INTEGER := 0;
        inst_en_ubr_flag : INTEGER := 0
        );
      port (
        inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        inst_b : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        inst_rnd : in std_logic_vector(2 downto 0);
        inst_DG_ctrl : in std_logic;
        z_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        status_inst : out std_logic_vector(7 downto 0)
        );
    end DW_fp_div_DG_inst;


architecture inst of DW_fp_div_DG_inst is

begin

    -- Instance of DW_fp_div_DG
    U1 : DW_fp_div_DG
    generic map ( sig_width => inst_sig_width, exp_width => inst_exp_width,
ieee_compliance => inst_ieee_compliance, faithful_round => inst_faithful_round,
en_ubr_flag => inst_en_ubr_flag )
    port map ( a => inst_a, b => inst_b, rnd => inst_rnd, DG_ctrl => inst_DG_ctrl, z =>
z_inst, status => status_inst );


end inst;

-- pragma translate_off
configuration DW_fp_div_DG_inst_cfg_inst of DW_fp_div_DG_inst is
 for inst
 end for; -- inst
end DW_fp_div_DG_inst_cfg_inst;
-- pragma translate_on
```

## HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_div_DG_inst( inst_a, inst_b, inst_rnd, inst_DG_ctrl, z_inst,
          status_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 0;
parameter faithful_round = 0;
parameter en_ubr_flag = 0;


input [sig_width+exp_width : 0] inst_a;
input [sig_width+exp_width : 0] inst_b;
input [2 : 0] inst_rnd;
input inst_DG_ctrl;
output [sig_width+exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_div_DG
    DW_fp_div_DG #(sig_width, exp_width, ieee_compliance, faithful_round, en_ubr_flag)
      U1 ( .a(inst_a), .b(inst_b), .rnd(inst_rnd), .DG_ctrl(inst_DG_ctrl), .z(z_inst),
.status(status_inst) );

endmodule
```

## Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

| Date | Release | Updates |
|---|---|---|
| April 2022 | DWBB_202203.1 | ■ Added the description of `status`[6] to Table 1-1 on page 1<br>■ Added the *en_ubr_flag* parameter to Table 1-2 on page 2 and to the examples on page 6 and page 7 |
| January 2022 | DWBB_202106.5 | ■ Adjusted description of the str implementation in Table 1-3 on page 3 |
| October 2020 | DWBB_202009.1 | ■ For enhanced NaN compatibility with the IEEE Std 754 standards, added a new value for *ieee_compliance* in Table 1-2 on page 2 |
| July 2020 | DWBB_201912.5 | ■ Adjusted the description of the *ieee_compliance* parameter in Table 1-2 on page 2<br>■ Added "Suppressing Warning Messages During Verilog Simulation" on page 5 |
| April 2020 | DWBB_201912.3 | ■ Updated the description of `rnd` in Table 1-1 on page 1 and *faithful_round* in Table 1-2 on page 2<br>■ For STAR 3124396, added a footnote to Table 1-2 on page 2 to update the error range when *faithful_round* = 1. This update is based on a limitation found during the investigation of STAR 3124396. Also, Table 1-5, "Error Ranges ($\varepsilon = z - a/b$)" was removed. |
| March 2019 | DWBB_201903.0 | ■ Added "Datapath Gating Control with DG_ctrl" on page 4<br>■ Clarified some information about minPower<br>■ Added this Revision History table and the document links on this page |

# Copyright Notice and Proprietary Information



### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com